# **Лабораторна робота №4**

Зворотний польський запис та обчислення математичного виразу

**Мета роботи:** Використовуючи теоретичне підґрунтя про зворотний польський запис розробити програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.

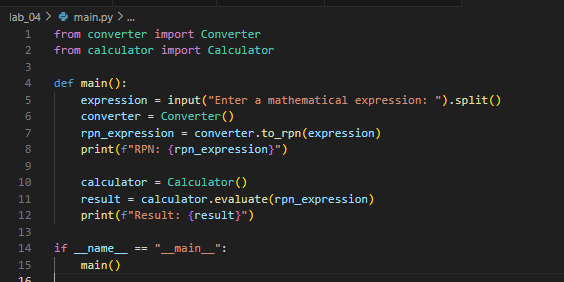
**Завдання до лабораторної роботи**

Використовуючи теоретичне відомості розробити програму яка на вхід отримує математичний вираз з довільною кількістю операндів, операторів та дужок. В першу чергу сформувати послідовність символів у ЗПН. На другому етапі виконання лабораторної роботи вирахувати результат послідовності, що була сформована, використовуючи алгоритм запису математичного виразу у ЗПН.

Текст програми разом зі звітом розмістити в директорії lab\_04. Директорію lab\_04 розмістити в директорії, що використовується для виконання практичних завдань по кожній лекції та має назву TP-KB-22[1 or 2]-Name-Surname.

Хід виконання завдання:

Файл main.py



Файл calculator.py



Файл converter.py



Вивід:



Текст Файлу main.py:

|  |
| --- |
| from converter import Converter  from calculator import Calculator  def main():  expression = input("Enter a mathematical expression: ").split()  converter = Converter()  rpn\_expression = converter.to\_rpn(expression)  print(f"RPN: {rpn\_expression}")  calculator = Calculator()  result = calculator.evaluate(rpn\_expression)  print(f"Result: {result}")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

Текст Файлу calculator.py:

|  |
| --- |
| class Calculator:  def evaluate(self, rpn\_expression):  stack = []  for token in rpn\_expression.split():  if token.isnumeric():  stack.append(int(token))  else:  right = stack.pop()  left = stack.pop()  if token == '+':  stack.append(left + right)  elif token == '-':  stack.append(left - right)  elif token == '\*':  stack.append(left \* right)  elif token == '/':  stack.append(left / right)  elif token == '^':  stack.append(left \*\* right)  return stack[0]  if \_\_name\_\_ == "\_\_main\_\_":  rpn\_expression = "3 4 2 \* 1 5 - 2 ^ / +"  calculator = Calculator()  result = calculator.evaluate(rpn\_expression)  print(f"Result: {result}") |

Текст Файлу converter.py:

|  |
| --- |
| class Converter:  def \_\_init\_\_(self):  self.stack = []  self.output = []  def precedence(self, op):  precedences = {'+': 1, '-': 1, '\*': 2, '/': 2, '^': 3}  return precedences.get(op, 0)  def to\_rpn(self, expression):  for token in expression:  if token.isnumeric():  self.output.append(token)  elif token == '(':  self.stack.append(token)  elif token == ')':  while self.stack and self.stack[-1] != '(':  self.output.append(self.stack.pop())  self.stack.pop()  else:  while self.stack and self.precedence(token) <= self.precedence(self.stack[-1]):  self.output.append(self.stack.pop())  self.stack.append(token)    while self.stack:  self.output.append(self.stack.pop())  return ' '.join(self.output)  if \_\_name\_\_ == "\_\_main\_\_":  expression = "3 + 4 \* 2 / ( 1 - 5 ) ^ 2".split()  converter = Converter()  rpn\_expression = converter.to\_rpn(expression)  print(f"RPN: {rpn\_expression}") |

Посилання на github:

<https://github.com/Lizarebenokk>